QuantNet: Learning to Quantize by Learning Within Fully Differentiable Framework

Junjie Liu[1], Dongchao Wen[1(✉)], Deyu Wang[1], Wei Tao[1], Tse-Wei Chen[2], Kinya Osa[2], and Masami Kato[2]

[1] Canon Information Technology (Beijing) Co., Ltd., Beijing, China
{liujunjie,wendongchao,wangdeyu,taowei}@canon-ib.com.cn
[2] Device Technology Development Headquarters, Canon Inc., Ota City, Japan
twchen@ieee.org

Abstract. Despite the achievements of recent binarization methods on reducing the performance degradation of Binary Neural Networks (BNNs), gradient mismatching caused by the Straight-Through-Estimator (STE) still dominates quantized networks. This paper proposes a meta-based quantizer named QuantNet, which utilizes a differentiable sub-network to directly binarize the full-precision weights without resorting to STE and any learnable gradient estimators. Our method not only solves the problem of gradient mismatching, but also reduces the impact of discretization errors, caused by the binarizing operation in the deployment, on performance. Generally, the proposed algorithm is implemented within a fully differentiable framework, and is easily extended to the general network quantization with any bits. The quantitative experiments on CIFAR-100 and ImageNet demonstrate that QuantNet achieves the significant improvements comparing with previous binarization methods, and even bridges gaps of accuracies between binarized models and full-precision models.

Keywords: Deep neural networks · Quantization · Compression

1 Introduction

Deep neural networks (DNNs) have achieved remarkable success in several fields in recent years. In particular, convolutional neural networks (CNNs) have shown state-of-the-art performance in various computer vision tasks such as image classification, object detection, trajectory tracking, etc. However, an increasing number of parameters in these networks also lead to the larger model size and higher computation cost, which gradually becomes great hurdles for many applications, especially on some resource-constrained devices with limited memory space and low computation ability.

To reduce the model size of DNNs, representative techniques such as network quantization [7,18,20,25,42], filters pruning [14,30,31], knowledge distillation [16,17,27] and deliberate architecture design [6,26] are proposed. As one of

A. Bartoli and A. Fusiello (Eds.): ECCV 2020 Workshops, LNCS 12539, pp. 38–53, 2020.
https://doi.org/10.1007/978-3-030-68238-5_4

typical solutions, the quantization based method quantizes floating-point values into discrete values in order to generate the quantized neural networks (QNNs) as compact as possible. In the most extreme case, if both network weights and network activations are binarized (BNNs) [8], the computation can be efficiently implemented via bitwise operations, which enables about 32× memory saving and 58× speeding up [33] on CPUs in inference.

Despite the advantages we mentioned above, how to alleviate performance degradation of quantized networks is still under research, especially for binarized networks. In general, BNNs involve a *sign* function to obtain signs of parameters. The non-differentiable sign function leads to gradient vanishing almost anywhere. To address this issue, some works [28,44] propose low-bit training algorithms to relieve the impact of gradients quantization errors, and another works focus on estimating the vanishing gradients. The Straight-Through Estimator (STE) [3] is commonly used to estimate the vanishing gradients during the back-propagation, while the well-known gradient mismatching problem [15,19,39] is introduced.

As the number of quantized bits decrease, the gradients estimated by STE depart further from the real gradients. Thus, the gradient mismatching is considered as the main bottleneck of performance improvements of binarized models. As one of promising solutions, estimating more accurate gradients is suggested by recent methods. Some of these methods [29,37,38,40] try to refine the gradients estimated by STE with extra parameters, and others [2,5] address the problem by replacing STE with learnable gradient estimators. Different from these efforts on estimating more accurate gradients, the individual method [25] employs a differentiable function *tanh* as a soft binarizing operation, in order to replace the non-differentiable function *sign*. *Thus, it will no longer require STE to estimate gradients.*

In this paper, we follow the idea of soft binarization, but we focus on solving two important issues that are left out. Firstly, although the soft binarization solves the problem of gradient mismatching, another issue of gradient vanishing from the function *tanh* arises. It not only causes the less ideal convergence behavior, but makes the solution highly suboptimal. Moreover, as the soft binarization involves a post-processing step, how to reduce the impact of the discretization errors on performance is very important. With these motivations, *we propose a meta-based quantizer named QuantNet for directly generating binarized weights with an additional neural network. The said network is referred to as a meta-based quantizer and optimized with the binarized model jointly.* In details, it not only generates the higher dimensional manifolds of weights for easily finding the global optimal solution, but also penalizes the binarized weights into sparse values with a task-driven priority for minimizing the discretization error. For demonstrating the effectiveness of our claims, we present the mathematical definition of two basic hypotheses in our binarization method, and design a joint optimization scheme for the QuantNet.

We evaluate the performance of our proposed QuantNet by comparing it with the existing binarization methods on the standard benchmarks of classification task with CIFAR-100 [23] and ImageNet [9]. As for the baseline with

different network architectures, AlexNet [24], ResNet [13], MobileNet [36] and DenseNet [11] are validated. The extensive experiments demonstrate that our method achieves remarkable improvements than state-of-the-arts across various datasets and network architectures.

In the following, we briefly review previous works related to network quantization in Sect. 2. In Sect. 3, we define the notations and present the mathematical definition of existing binarization methods. For Sect. 4, we present two basic hypotheses of our binarization method and exhibit the implementation details. Finally, we demonstrate the effectiveness and efficiency of our method in Sect. 5, and make the conclusions in Sect. 6.

2 Related Work

In this section, we briefly review existing methods on neural network quantization. As the most typical strategy to achieve the purpose of network compression, the network quantization has two major benefits - reducing the model size while improving the inference efficiency. Comparing with the strategies of network filters pruning [14,30,31] and compact architecture design [6,26], how to alleviate the performance degradation in quantized model [19] is still unsolved, especially for the binarized model [1,15,39].

Deterministic Weight Quantization. Through introducing a deterministic function, traditional methods quantize network weights (or activations) by minimizing quantization errors. For examples, BinaryConnect [8] uses a stochastic function for binarizing weights to the binary set $\{+1, -1\}$, which achieves better performance than two-step approachs [12,21] on several tasks. Besides, XNOR-net [33] scales the binarized weights with extra scaling factors and obtains better results. Furthermore, Half-Wave-Gaussian-Quantization (HWGQ) [4] observes the distribution of activations, and suggests some non-uniform quantization functions for constraining unbounded values of activations. Instead of binarizing the model, the ternary-connect network [43] and DoReFa-Net [42] perform the quantization with multiple-bits via various functions to bound the range of parameters.

These methods purely focus on minimizing quantization errors between full-precision weights and quantized weights, however less quantization errors do not necessarily mean better performance of a quantized model.

Loss-Aware Weight Quantization. As less quantization errors do not necessarily mean better performance of a quantized model, several recent works propose the loss-aware weight quantization in terms of minimizing the task loss rather than quantization errors. The loss-aware binarization (LAB) [18] proposes a proximal Newton algorithm with diagonal Hessian approximation that minimizes the loss with respect to the binarized weights during optimization. Similar to LAB, LQ-Net [41] allows floating-point values to represent the basis of quantized values during the quantization. Besides, PACT [7] and SYQ [10] suggest parameterized functions to clip the weights or activation value during training.

In the latest works, QIL [20] parameterizes the non-linear quantization intervals and obtains the optimal solution by minimizing with the constraint from task loss. And self binarization [25] employs a soft-binarization function to evolve weights and activations during training to become binary.

In brief, through introducing learnable constraints or scaling factors, these methods alleviate the performance degradation in their quantized models, but the gradient mismatching problem caused by the sign function and STE [1] is still unconsidered.

Meta-based Weight Quantization. As the quantization operator in training process is non-differentiable, which leads to either infinite gradients or zero gradients, MixedQuant [37] addresses a gradient refiner by introducing the assistant variable for approximating the more accurate gradients. Similar to MixedQuant, ProxQuant [2] proposes an alternative approach that formulates quantized network training as a regularized learning problem and optimizes it by the proximal gradients. Furthermore, Meta-Quant [5] proposes a gradient estimator to directly learn the gradients of quantized weights by a neural network, in order to remove STE commonly used in back-propagation.

Although such methods have noticed that refining the gradients computed by STE or directly estimating the gradients by meta-learner is helpful to alleviate the problem of gradient mismatching, the increasing complexity of learning gradients introduces a new bottleneck.

3 Preliminaries

Notations. For a vector x, where $x_i; i \leq n$ is the element of x, we use $\sqrt{x}$ to denote the element-wise square root, $|x|$ denotes the element-wise absolute value, and $\|x\|_p$ is the p-norm of x. $sign(x)$ is an element-wise function denoting that $sign(x_i) = 1; \forall i \leq n$ if $x_i \geq 0$ and -1 otherwise. We use $diag(X)$ to return the diagonal elements of matrix X, and $Diag(x)$ to generate a diagonal matrix with vector x. For two vectors x and y, $x \odot y$ denotes the element-wise multiplication and $x \oslash y$ denotes the element-wise division. For a matrix X, $vec(X)$ denotes to return a vector by stacking all the columns of X. In general, ℓ is used to denote the objective loss, and both $\partial \ell / \partial x$ and $\nabla \ell(x)$ denote the derivative of ℓ with respect to x.

Background of Network Binarization. The main operation in network binarization is the linear (or non-linear) discretization. Taking a multilayer perception (MLP) neural network as an example, one of its hidden layers can be expressed as

$$\mathbf{w}_q = f(\mathbf{w})^r \odot binarize(\mathbf{w}) \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^{m \cdot n}$ is the full-precision weights, and m, n are the number of input filter channels, the number of output filter channels[1], respectively. Based

[1] In this paper, the kernels on full connected layer are regarded as a special type of the convolutional kernels.

on the full-precision (floating-point) weights, the corresponding binarized weights $\mathbf{w}_q$ is computed by two separate functions $f(\mathbf{w})^r$ and $binarize(\mathbf{w})$, and the goal is to represent the floating-point elements in $\mathbf{w}$ with one bit.

In BinaryConnect [8], $f(\mathbf{w})^r$ is defined as the constant 1, and $binarize(\mathbf{w})$ is defined by $sign(\mathbf{w})$, which means each element of $\mathbf{w}$ will be binarized to {-1,+1}. For XNOR-Net [33], it follows the definition of BinaryConnect on $binarize(\mathbf{w})$, but further defines $f(\mathbf{w}_t)^r = \|\mathbf{w}_t\|_1/(m \times n)$ and r is defined as the constant 1, where t is the current number of training iterations. Different from the determining function on $f(\mathbf{w})$, the Loss-Aware Binarization (LAB) [18] suggests a task-driven $f(\mathbf{w}_t)$ with the definition of $\|d_{t-1} \odot \mathbf{w}_t\|_1/\|d_{t-1}\|_1$, where d_{t-1} is a vector containing the diagonal $diag(D_{t-1})$ of an approximate Hessian D_{t-1} of the task loss. Furthermore, QIL [20] extends $f(\mathbf{w}_t)^{r_t}$ into a nonlinear projection by setting r_t to be learnable and $r_t > 1$ for all t. Considering the back-propagation, as STE [39] with $sign$ function introduces the major performance bottleneck for BNNs, Self-Binarization [25] defines $binarize(\mathbf{w})$ as $tanh(\mathbf{w})$. In the training of BNNs, the $tanh$ function transforms the full-precision weights $\mathbf{w}$ to obtain weights $\mathbf{w}_q$ that are bounded in the range $[-1, +1]$, and these weights are closer to binary values as the training converges. After the training, $\mathbf{w}_q$ are very close to the exact set of $\{+1, -1\}$, and the fixed point values will be obtained by taking the sign of the $\mathbf{w}_q$.

4 Methodology

In this section, we firstly present the mathematical definition of two basic hypotheses in our binarization method. Then we propose a meta-based quantizer named QuantNet for directly generating binarized weights within a fully differentiable framework. Moreover, a joint optimization scheme implemented in a standard neural network training is designed to solve the proposal.

4.1 Assumptions

As can be seen, the work [25] replaces the hard constraint $sign$ with the soft penalization $tanh$, and penalizes the output of $tanh$ to be the closest binary values. However, there are two important issues which are ignored.

In the case of binarizing weights with $tanh$, as most of the elements in binarized weights are close to $\{+1, -1\}$ at the early stage of training, these elements will reach saturation simultaneously, and then cause the phenomenon of gradients vanishing. In brief, if the element is saturated on $+1$, it will not be able to get close to -1 again. On the contrary, the case of the element saturated on -1 is the same. It means that flipping values of these elements is impossible. As a result, only a few unsaturated elements will oscillate around zero, which causes the less ideal convergence behavior and makes the solution highly suboptimal.

Moreover, different from the hard constraint methods, the soft penalization method contains a post-processing step with rounding functionality, and it rounds the binarized weights for further obtaining the fixed point (discrete)

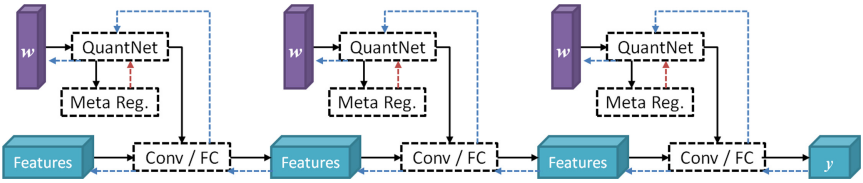

Fig. 1. The architecture of our binarization method. The solid and dashed (blue color) lines represent the feed-forward process and the gradient flow of back-propagation separately, and the dashed line with red color means the meta-gradient flow from the meta regularization (Best viewed in color). (Color figure online)

values. With the increasing number of the network parameters, the discretization error caused by the rounding function will be the major factor to limit performances. To propose our method for solving above issues, we make two fundamental hypotheses in the following.

Assumption 1: We assume that there exists the functional $\mathcal{F}$ to form $tanh(\mathcal{F}(\mathbf{w}))$, and $\lim_{\mathbf{w}\to\infty} \nabla\mathcal{F}(\mathbf{w}) \cdot (1 - tanh^2(\mathcal{F}(\mathbf{w})) \neq 0$, then the derivative of $tanh()$ with respect to $\mathbf{w}$ is expressed as

$$\begin{aligned} &\lim_{\mathbf{w}\to\infty} \frac{\partial tanh(\mathcal{F}(\mathbf{w}))}{\partial \mathbf{w}} \neq 0 \\ w.r.t. \quad &\nabla tanh(\mathcal{F}(\mathbf{w})) = \frac{\partial \mathcal{F}(\mathbf{w})}{\partial \mathbf{w}} \Big(1 - tanh^2(\mathcal{F}(\mathbf{w}))\Big) \end{aligned} \tag{2}$$

Assumption 1 derives a corollary that if $\mathbf{w}$ is out of a small range like $[-1, +1]$, the gradient of $tanh(F())$ for $\mathbf{w}$ will not severely vanish. Through generating the higher dimensional manifolds of full-precision weights $\mathbf{w}$, the gradient vanishing during optimization is relieved, which allows optimizers to solve the globally optimal.

Assumption 2: We assume for a vector $v \in \mathbb{R}^n$ that is k-sparse, there exists an extremely small $\epsilon \in (0, 1)$ with optimal $\mathbf{w}_q^*$, in the optimization of $\ell(\mathbf{w}_q)$ with the objective function ℓ, it has the property that

$$\begin{aligned} &\lim_{\mathbf{w}_q \to \mathbf{w}_q^*} \|\ell(\mathbf{w}_q) - \ell(sign(\mathbf{w}_q))\|_2^2 = 0 \\ s.t. \quad &(1 - \epsilon) \leq \frac{\ell(\mathbf{w}_q^* v)}{\ell(\mathbf{w}_q^*)} \leq (1 + \epsilon) \end{aligned} \tag{3}$$

Assumption 2 derives a conclusion that if the said constraint of $\frac{\ell(\mathbf{w}_q^* v)}{\ell(\mathbf{w}_q^*)}$ with ϵ is satisfied, it represents the top-k elements in $\mathbf{w}_q^*$ dominate the objective function ℓ, while the remaining elements do not affect the output seriously. In this case, the discretization error caused by the post-processing step is minimized, as the

sign of top-k elements are equal to themselves. In brief, the optimization no longer requires all elements in $\mathbf{w}_q$ to converge to $\{+1, -1\}$ strictly, but penalizes it to satisfy the top-k sparse with the task-driven priority.

4.2 Binarization with the QuantNet

Based on above two fundamental hypotheses, we propose a meta-based quantizer named QuantNet for directly generating the binarized weights. Our proposal is to form the functional $\mathcal{F}$ for transforming $\mathbf{w}$ into higher dimensional mainfold $\mathcal{F}(\mathbf{w})$, and optimizing the dominant elements $\mathbf{w}_q$ to satisfy the sparse constraint.

As for implementation details of QuantNet, we design an encoding module accompanied by a decoding module, and further construct an extra compressing module. Specially, suppose full-precision weights come from a convolution layer with 4D shape $\mathbb{R}^{k\times k\times m\times n}$, where k, m and n denote the kernel size, the number of input channels and the number of output channels, respectively.

The input weights will be firstly reshaped into the 2D shape $\mathbb{R}^{m\cdot n\times k^2}$. It means that QuantNet is a kernel-wise quantizer and process each kernel of weights independently, where the batch size is the number of total filters of full-precision weights. In the encoding and decoding process, it firstly expands the reshaped weights into higher dimensional mainfold, which is achieved with the dimensional guarantee that makes the output shape of encoding module to satisfy $\mathbb{R}^{m\cdot n\times d^2}$, $s.t.\ d \gg k$. And then, the compressing module is to transform the higher dimensional manifolds into low-dimensional spaces. If the manifold of interest remains non-zero volume after the compressing process, it corresponds to a higher priority to improve the performance of binarized model on the specific task. Finally, the decoding module generates the binarized weights with the output of the compressing module and the soft binarization function, while restoring the original shape of full-precision weights for main network optimization.

The Fig. 1 provides visualization of QuantNet in the architecture.

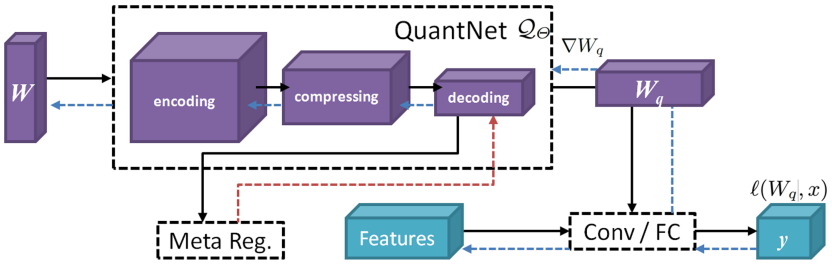

Fig. 2. The architecture of QuantNet. QuantNet $\mathcal{Q}_\Theta$ takes the full-precision weights W as the input, and directly outputs the binarized weights W_q for the network. With the loss of $\ell(W_q, x)$, W_q will be directly updated in the back-propagation, and ∇W_q will be used to update Θ in QuantNet $\mathcal{Q}$. Finally, a new W is computed during this training step.

Feed-Forward Step. Given a full-precision weights[2] W, the proposed QuantNet $\mathcal{Q}_\Theta$ incorporates the parameters Θ to generate the binarized weights W_q with $tanh(\mathcal{Q}_\Theta(W))$. After W is quantized as W_q, the loss ℓ is generated by $\ell(W_q, \{x, y\})$ with the training set {x, y} (Fig. 2).

Back-Propagation Step. The gradient of ℓ with regard to W_q in each layer is computed by the back-propagation. For example, the gradient of weights g_{W_q} in last layer is computed by $\nabla\ell(W_q)$. Then, the QuantNet $\mathcal{Q}_\Theta$ receives the g_{W_q} from the corresponding W_q, and updates its parameters Θ by

$$g_\Theta = \frac{\partial \ell}{\partial W_q}\frac{\partial W_q}{\partial \Theta} = g_{W_q}\frac{\partial W_q}{\partial \Theta} \tag{4}$$

and the gradients of g_Θ is further used to update the full-precision weights W by,

$$W^{t+1} = W^t - \eta \cdot g_\Theta \tag{5}$$

where t denotes the t-th training iteration and η is the learning rates defined for QuantNet.

In practice, QuantNet is applied layer-wise. However, as the number of extra parameters introduced by QuanNet is much less than the network weights, so the computation cost caused by our proposal is acceptable during the network training as shown in the Table 5.

4.3 Optimization

As for the optimization of QuantNet, it is included in the target network that will be binarized. Given the full-precision weights, QuantNet generates the binarized weights to apply on objective tasks, and it is optimized with the back-propagation algorithm to update all variables. In brief, our binarization framework is fully differentiable without any gradient estimators, so there is no information loss during the binarization. We now present the optimization details.

QuantNet Optimization. For satisfying the constraint in Assumption 2, we propose an objective function inspired by the idea of sparse coding. It constrains the compressing process in QuantNet during the binarization. Let $\mathbf{w}_q$ be the binarized weights and introduce a reference tensor $\mathbf{b}$, we aim to find an optimal $\mathbf{w}_q^*$ that satisfies

$$\mathbf{w}_q^* = \arg\min_{\mathbf{w}_q} \|\mathbf{b} - \sqrt{\mathbf{w}_q^2}\|_2 + \|\mathbf{w}_q\|_1, \ s.t.\ \mathbf{b} \in \{1\}^{m\cdot n} \tag{6}$$

where, tensor $\mathbf{b}$ is chosen to be all ones to make elements in $\mathbf{w}_q$ to get close to -1 or +1. As Eq. 6 is independent of the task optimization of binarized models, we alternately solve it with an extra optimizer during the standard network training. At each iteration of optimization, there is adversarial relationship between Eq. 4 and Eq. 6, and the optimization tries to find the balance between minimizing

[2] We omit the notation of layers l in W_l for simplification.

Algorithm 1: Generating the binarized weights with QuantNet

Input: the full-precision weights W, the QuantNet $\mathcal{Q}$ with parameters Θ, and the training set {X, Y}, training iteration t, $\epsilon = 1e-5$
Output: the optimally binarized weights W_q^*
Training for each layer
for $t = 0; t \leq T$ **do**
  Feed-Forward;
    Compute W_q^t with $tanh(\mathcal{Q}_{\Theta^t}(W^t))$;
    Compute $\ell(W_q^t, \{x^t, y^t\})$ with W_q^t and $\{x^t, y^t\}$;
  Back-Propagation;
    Compute ∇W_q^t with $\ell(W_q^t, \{x^t, y^t\})$;
    Compute $\nabla \Theta^t$ with Eq. 4 and Eq. 6;
    Update the W^t with Eq. 5;
end
Discretization Step
$W_q^* = sign(W_q^T)$;

the binarization error while penalizing the sparsity of binarized weights based on the task priority.

Binarized Model Optimization. As for the optimization of binarized model, we use the standard mini-batch based gradient descent method. After the QuantNet $\mathcal{Q}_\Theta$ is constructed and initialized, the QuantNet optimization is accompanied with the training process of the binarized model, and the objective function of binarized model optimization depends on the specific task. With the specific objective function, the task-driven optimizer is employed to compute the gradients for each binarized layer. The operations for whole optimization are summarized in Algorithm 1.

5 Experiments

In this section, we firstly show the implementation details of our method and experiment settings. Secondly, the performance comparison between our method and STE-based or Non STE-based methods is generated, which further includes the analysis of convergence behaviour.

5.1 Implementation Details

As for the implementation details in experiments, we run each experiment five times with the same initialization function from different starting points. Besides, we fix the number of epoches for training and use the same decay strategy of learning rate in all control groups. At the end, we exhibit the average case of training loss and corresponding prediction accuracy. We show the implementation details as follows.

Network Architecture. We apply the unit with structure of "FC-BN-Leaky ReLU" to construct QuantNet, and each processing module in QuantNet contains at least one unit. For reducing the complexity of network training, QuantNet used in experiments contains only one unit for each processing module, and we still observe a satisfied performance during evaluation. Similar to the existing methods [33,37,41], we leave the first and last layers and then binarizing the remaining layers. For comparison, a fully binarized model by binarizing all layers is also generated. Considering that the bitwise operations can speedup the inference of network significantly, we analyze the balance between the computation cost saving and model performance boosting by these two models. The experiment result exhibits only 0.6% accuracy drop (more than 1% in previous methods) in CIFAR-10 [23] with ResNet-20, in the case that all layers are binairzed.

Initialization. In experiments, all compared methods including our method use the truncated Gaussian initialization if there is not specified in their papers, and all binarized model from experiments are trained from scratch without leveraging any pre-trained model. As for the initialization of QuantNet, we employ the normal Gaussian initialization for each layer. Furthermore, we also evaluate the random initialization, which initialize the variable with the different settings of the mean and variance, but there is not significant difference on the results.

Hyper-parameters and Tuning. We follow the hyper-parameter settings such as the learning rate, batch size, training epoch and weight decay of their original paper. For fair comparison, we use the default hyper-parameters in Meta-Quant [5] and Self-Binarizing [25] to generate the fully binarized network. As for the hyper-parameters of our QuantNet, we set the learning rate as $1e-3$ and the moving decay factor as 0.9. We also evaluate different optimization methods including SGD(M) [35], Adam [22] and AMSGrad [34]. Although we observe that the soft binarization in AMSGrad has a faster convergence behaviour than the others, we still use the SGD(M) with average performance for all methods to implement the final comparison. In future, we plan to analyze the relationship between the soft binarization and different optimizers (Table 1).

Table 1. Comparison with different optimizer on ResNet-20 for CIFAR10.

Optimizer	Accuracy (%)	Training time
SGD(M)	90.04	1.0×
Adam	89.98	˜1.2×
AMSGrad	90.12	˜0.9×

5.2 Performance Comparison

QuantNet aims at generating the binarized weights without STE and other estimators. Hence, we compare it with both STE-based binarization methods

[8,18,20,41,42] and Non STE-based binarization methods [2,5,25,37] with the idea of avoiding the discrete quantization. In details, the evaluation is based on the standard benchmark of classification task with CIFAR-100 [23] and ImageNet [9], and the base network architectures are based on the AlexNet [24], ResNet [13] and MobileNet [36].

Evaluation of the Discretization Error. As the soft binarization method [25] always involves a post-processing step, which aims at transforming the float-point weights into the fixed-point weights, we name this step the discretization step which is shown in Algorithm 1. For comparing the discretization error caused by the step between the self-binarizing [25] and the proposed QuantNet, we generate both two binarized models for self-binarizing and QuantNet, and use the notation (D) to denote the prediction accuracy of binarized model after the discretization step, which means the weights in the binarized model is transformed into the integer exactly. As shown in the Table 2, QuantNet achieves the best performance even better than the FP, the major reason is that the sparse constraint encourages a better generalization ability. Moreover, since the discretization error is considered in our algorithm during the binarizing process, comparing to the accuracy drop 1.85% in self-binarizing [25], QuantNet only reduces 0.59%.

Table 2. Prediction accuracy of binarized AlexNet on CIFAR-10. The FP represents the full-precision model with 32-bits for both weights and activations.

Methods	Bit-width (W/A)	Acc.(%)	Acc.(%) (after discretization)
FP	32/32	86.55	–
Self-Binar. [25]	1/1	86.91%	84.31%
Ours	1/1	**87.08**%	**86.49**%

Comparison with STE-Based Binarization. We evaluate our QuantNet with the STE-based binarization methods, and report the top-1 accuracy in Table 3. Besides, we use PACT [7] to quantize the activation into 2 bits if the compared method does not support the activation quantization. For the compared prediction accuracy used in this table, we use the results from the original paper if it is specified. Overall, QuantNet achieves the best performance compared to existing STE-based methods, which surpasses QIL [20] more than 2% before the discretization step, and even obtain a comparable performance with the full-precision model. It demonstrates the advantage of directly binarizing the weights within a fully differentiable framework. Although the discretization (rounding operation) introduces a post-processing step, the experiment results still prove the effectiveness of our binarization method, and the degradation of prediction accuracy caused by rounding is negligible.

Comparison with Non STE-Based Binarization. As for the Non STE-based binarization methods, it mainly includes two categories: *learning better*

Table 3. Top-1 accuracy (%) on CIFAR-100. Comparison with the existing methods on ResNet-56 and ResNet-110.

Method	Bit-width	ResNet-56	ResNet-110
FP	32W32A	71.22%	72.54%
BWN [8]	1W2A	64.29%	66.26%
DoReFa-Net [42]	1W2A	66.42%	66.83%
LAB [18]	1W2A	66.73%	67.01%
LQ-Nets [41]	1W2A	66.55%	67.09%
QIL [20]	1W2A	67.23%	68.35%
Ours	1W2A	**69.38**%	**70.17**%
Ours(D)	1W2A	**68.79**%	**69.48**%

gradients for non-differentiable binarization function, and *replacing the non-differentiable function with the differentiable one*. We compare the QuantNet with the representative works in these two categories - ProxQuant [2] and Meta-Quant [5] in the first and Self-Binarizing [25] in the second. With the increasing number of parameters in larger architecture [13], although the methods [2,5] related gradient refinement have improved the performance effectively, the bottleneck caused by the gradient estimation appears obviously, and our method have achieved the significant improvement than these methods (Table 4). Moreover, as the discretization error caused by the rounding operation is well considered by our method, QuantNet is affected less than Self-Binarizing [25].

Table 4. Top-1 accuracy (%) on ImageNet. Comparison with the existing methods on AlexNet (left), ResNet-34 (right).

Method	Bit-width	AlexNet
FP	32W32A	55.07%
Self-Binar. [25]	1W32A	52.89%
Self-Binar.(D) [25]	1W32A	50.51%
Ours	1W32A	**54.06**%
Ours(D)	1W32A	**53.59**%

Method	Bit-width	ResNet-34
FP [32]	32W32A	73.30%
ProxQuant [2]	1W32A	70.42%
Meta-Quant [5]	1W32A	70.84%
Ours	1W32A	**71.97**%
Ours(D)	1W32A	**71.35**%

Convergence Analysis. We analyze the convergence behaviour of our QuantNet and other binarization methods during the training process. In details, we use ResNet-34 as the base architecture, and compare with the STE-based methods and non-STE based methods separately. For the first case, QuantNet exhibits a significantly smooth loss curve over STE, including much faster convergence speed and lower loss values, and it also achieves the best prediction accuracy in

the test reported in Table 3. The main reason of the better convergence of our method is that QuantNet is totally differentiable during the optimization. Furthermore, we analyze the proposed method in the second case, and we observe that all the non-STE based methods can smooth the loss curve effectively, but our method achieve the lowest loss value as there is not estimation of gradients.

Complexity of Models. As our QuantNet involves the extra computation cost and parameters during the optimization, we analyze its efficiency comparing to the traditional STE-based methods and other Meta-based methods. For QuantNet, it is independent of the scale of input resource, and its time complexity is related to the amount of its parameters. In the Table 5, the total training time cost is exhibited, and we leave the inference step since the QuantNet is removed in this step. For the setting of experiment in this table, the base architecture ResNet-34 is used, and the bitwise operation is not implemented for all cases.

Table 5. Training time on ResNet-34

Bit-width (1W/2A)	Training time(iter./s)
FP(32W/32A)	1.0×
MixedQuant [37]	1.5×
Meta-Quant [5]	2.8×
Self-Binar. [25]	1.2×
Ours	1.7×

6 Conclusions

In the paper, we present a meta-based quantizer QuantNet to binarize the neural network, which directly binarize the full-precision weights without STE and any learnable gradient estimators. In contrast to the previous soft binarizing method, the proposed QuantNet not only solves the problem of gradient vanishing during the optimization, but also alleviates the discretization errors caused by the post-processing step for obtaining the fixed-point weights. The core idea of our algorithm is to transform the high dimensional manifolds of weights, while penalize the dominant elements in weights into sparse according to the task-driven priority. In conclusion, the QuantNet outperforms the existing binarization methods on the standard benchmarks, which not only can be applied on weights, but also can be extended to activations (or quantization with other bits) easily.

References

1. Alizadeh, M., Fernández-Marqués, J., Lane, Nicholas, D., Gal, Y.: An empirical study of binary neural networks' optimisation. In: International Conference on Learning Representations (ICLR) (2019)
2. Bai, Y., Wang, Y.X., Liberty, E.: Quantized neural networks via proximal operators. In: International Conference on Learning Representations (ICLR) (2019)
3. Bengio, Y., Leonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)

4. Cai, Z., He, X., Jian, S., Vasconcelos, N.: Deep learning with low precision by half-wave gaussian quantization. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
5. Chen, S., Wang, W., Pan, S.J.: MetaQuant: learning to quantize by learning to penetrate non-differentiable quantization. In: Annual Conference on Neural Information Processing Systems (NIPS) (2019)
6. Cheng, Y., Wang, D., Zhou, P., Zhang, T.: A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282 (2017)
7. Choi, J., Wang, Z., Venkataramani, S., Chuang, P.I.J., Srinivasan, V., Gopalakrishnan, K.: Pact: Parameterized clipping activation for quantized neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
8. Courbariaux, M., Bengio, Y., David, J.P.: BinaryConnect: training deep neural networks with binary weights during propagations. In: Annual Conference on Neural Information Processing Systems (NIPS), pp. 3123–3131 (2015)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F.: ImageNet: a large scale hierarchical image database. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
10. Faraone, J., Fraser, N., Blott, M., Leong, P.H.: SYQ: learning symmetric quantization for efficient deep neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
11. Gao, H., Zhuang, L., Laurens, V.D.M.: Densely connected convolutional networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
12. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. In: International Conference on Learning Representations (ICLR) (2016)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
14. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: The IEEE International Conference on Computer Vision (ICCV), pp. 1389–1397 (2017)
15. Helwegen, K., Widdicombe, J., Geiger, L., Liu, Z., Kwang-Ting, C., Nusselder, R.: Latent weights do not exist: rethinking binarized neural network optimization. In: Advances in Neural Information Processing Systems (NIPS) (2019)
16. Heo, B., Lee, M., Yun, S.: Knowledge distillation with adversarial samples supporting decision boundary. In: Association for the Advance of Artificial Intelligence (AAAI) (2019)
17. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. Comput. Sci. **14**(7), 38–39 (2015)
18. Hou, L., Yao, Q., Kwok, J.T.: Loss-aware binarization of deep networks. In: International Conference on Learning Representations (ICLR) (2017)
19. Hou, L., Zhang, R., Kwok, J.T.: Analysis of quantized models. In: International Conference on Learning Representations (ICLR) (2019)
20. Jung, S., et al.: Learning to quantize deep networks by optimizing quantization intervals with task loss. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4350–4359 (2019)
21. Kin, Y.D., Park, E., Yoo, S., Choi, T., Yang, L., Shin, D.: Compression of deep convolutional neural networks for fast and low power mobile applications. In: International Conference on Learning Representations (ICLR) (2016)

22. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
23. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Master's thesis (2009)
24. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS) (2012)
25. Lahoud, F., Achanta, R., Márquez-Neila, P., Süsstrunk, S.: Self-binarizing networks. In: arXiv preprint arXiv:1902.00730 (2019)
26. Li, D., Wang, X., Kong, D.: DeepRebirth: accelarating deep neural network execution on mobile device. In: International Conference on Learning Representations (ICLR) (2017)
27. Liu, J., et al.: Knowledge representing: Efficient, sparse representation of prior knowledge for knowledge distillation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2019)
28. Liu, J., et al.: BAMSProd: a step towards generalizing the adaptive optimization methods to deep binary model. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2020)
29. Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., Cheng, K.-T.: Bi-real net: enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11219, pp. 747–763. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01267-0_44
30. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: The IEEE International Conference on Computer Vision (ICCV), pp. 2736–2744 (2017)
31. Luo, J.H., Wu, J., Lin, W.: ThiNet: a filter level pruning method for deep neural network compression. In: Advances in Neural Information Processing Systems (NIPS), pp. 5068–5076 (2017)
32. Qin, H., et al.: Forward and backward information retention for accurate binary neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
33. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: ImageNet classification using binary convolutional neural networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 525–542. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_32
34. Reddi, S.J., Kale, S., Kumar, S.: On the convergence of Adam and beyond. In: International Conference on Learning Representations (ICLR) (2018)
35. Robbins, H., Monro, S.: A stochasitc approximation method. Ann. Math. Stat. **22**, 400–407 (1951)
36. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: inverted residuals and linear bottlenecks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520 (2018)
37. Uhlich, S., et al.: Mixed precision DNNs: all you need is a good parametrization. In: International Conference on Learning Representations (ICLR) (2020)
38. Wang, P., Hu, Q., Zhang, Y., Zhang, C., Liu, Y., Cheng, J.: Two-step quantization for low-bit neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
39. Yin, P., Lyu, J., Zhang, S., Osher, S., Qi, Y., Xin, J.: Understanding straight-through estimator in training activation quantized neural nets. arXiv preprint arXiv:1903.05662 (2019)

40. Yin, P., Zhang, S., Lyu, J., Osher, S., Qi, Y., Xin, J.: Blended coarse gradient descent for full quantization of deep neural networks. arXiv preprint arXiv:1808.05240 (2018)
41. Zhang, D., Yang, J., Ye, D., Hua, G.: LQ-Nets: learned quantization for highly accurate and compact deep neural networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11212, pp. 373–390. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01237-3_23
42. Zhou, S., Ni, Z., Zhou, X., He, W., Zou, Y.: DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160 (2016)
43. Zhu, C., Han, S., Mao, H., Dally, W.J.: Trained ternary quantization. In: International Conference on Learning Representations (ICLR) (2016)
44. Zhu, F., et al.: Towards unified INT8 training for convolutional neural network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)